REPORT DOCUMENTATION PAGE

AFRL-SR-AR-TR-04-

0290

1. REPORT DATE (DD-MM-YYYY) 26-05-2004	2. REPORT TYPE Final Project Summary Report	3. DATES COVERED (From - To) From: 01-08-2000 – To: 31-12-2003

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBERS
Capturing and Exploiting Semantic Relationships for Information Management	5b. GRANT NUMBER F49620-00-1-0337
	5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)	5d. PROJECT NUMBER
Yolanda Gil	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) USC/INFORMATION SCIENCES INSTITUTE 4676 ADMIRALTY WAY MARINA DEL REY, CA 90292-6695	8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research 4015 Wilson Blvd. Arlington, VA 22203	10. SPONSORING/MONITOR'S ACRONYM(S) AFOSR
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER

12. DISTRIBUTION AVAILABILITY STATEMENT
Distribution Statement A. Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

TRELLIS provides an interactive environment that allows users to add their observations, opinions, and conclusions as they analyze information by making semantic annotations to documents and other on-line resources. Our work concentrated on four major areas: 1) designing a vocabulary to annotate information analysis, 2) creating semantic markup representations for this vocabulary and annotating analysis products, 3) deriving trust ratings for sources used in collections of analyses, and 4) developing the TRELLIS interface and tools.

15. SUBJECT TERMS

intelligence analysis, argumentation, semantic markup, intelligent user interfaces, interactive decision aids

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Yolanda Gil
a. REPORT	b. ABSTRACT	c. THIS PAGE			
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	55	19b. TELEPHONE NUMBER (Include area code) (310) 448-8794

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

20040706 079

TRELLIS:
Capturing and Exploiting Semantic Relationships
for Information and Knowledge Management

Award Number F49620-00-1-0337

Final Technical Report

Yolanda Gil (PI)
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
(310) 822-1511
gil@isi.edu

TRELLIS:
Capturing and Exploiting Semantic Relationships for Information and Knowledge Management

Abstract

TRELLIS provides an interactive environment that allows users to add their observations, opinions, and conclusions as they analyze information by making semantic annotations to documents and other on-line resources. Our work concentrated on four major areas: 1) designing a vocabulary to annotate information analysis, 2) creating semantic markup representations for this vocabulary and annotating analysis products, 3) deriving trust ratings for sources used in collections of analyses, and 4) developing the TRELLIS interface and tools.

1 Project Overview

In a world of overwhelming on-line information access and global communications, more and more people are asked to provide faster and more accurate answers based on up-to-date knowledge that is increasingly more disseminated in vast amounts of information sources. The problem is especially acute in the world of intelligence analysis, the proposed application area for this work.

There are two key research objectives in this work. First, we propose to develop techniques to capture and exploit semantic interrelationships among information items. Our approach will be to use a semantic markup vocabulary that will enable users to specify semantic annotations not only the information items themselves but also the links that users make to relate individual items. We will provide a core vocabulary that will contain both general and domain-specific terms, and that will be extensible by users. We will develop tools that will analyze and exploit these annotations to support users in further analysis, sharing, and integration.

A second objective of this research is to support users in creating new knowledge fragments from raw information sources and from other fragments. The key to our approach is to use the semantic annotations to capture the derivation and rationale of their answers to stated questions as they progressively generate new knowledge based on their expertise and viewpoint. Capturing this information results in significant added value to the original raw information sources. We will support users to highlight key salient information from large reports and documents, to add new intermediate knowledge fragments based on their analysis and integration of existing information, and to finally put together these fragmented pieces into an overall picture of the situation.

This work is motivated by our previous research on knowledge acquisition within the EXPECT project. In order for non-programmers to add knowledge into a system, they need to be guided step by step through the modeling and knowledge representation decisions that knowledge engineers normally do. Users need to be guided through several steps:
1) collecting relevant documents and other information and data sources,
2) analyzing, grouping, and indexing related information,
3) relating the information into structured and consistent form, and
4) formalizing the knowledge into a logic formalism that supports automated inference and reasoning.

In our approach, a user could use semantic annotations to specify how each piece of knowledge comes about as they follow each of these steps. The results of each step would remain part of the knowledge base, so the rationale for each piece of knowledge in the system is captured. This approach would be very useful to maintain the knowledge base and to integrate it with other reasoning modules, since the source and rationale for each piece of knowledge will be available in the knowledge base itself instead of disappearing with the knowledge engineers that created it.

The ultimate scientific goal of the project is to contribute to the vision of a Semantic Web that has been put forward by the World Wide Web consortium.

Our approach has significant implications and benefits for the management of knowledge assets that many companies and government institutions are beginning to practice. Making documents available on-line and providing indexing and keyword search are a good first step, but our approach would support more ambitious information processing and management than ever before. By providing increasingly more structure to on-line information, as well as means to customize the way the information is organized, our approach would enable the development of intelligent information management systems that can process and retrieve information in ways specified by the end users themselves. This would result in a new generation of knowledge management, sharing, and dissemination systems.

In summary, the goal of TRELLIS is to provide an interactive environment that allows users to add their observations, opinions, and conclusions as they analyze information by making semantic annotations to documents and other on-line resources. This is in essence a knowledge acquisition problem, where the user is adding new knowledge to the system based on their expertise as they analyze information.

In previous work within the EXPECT project, we have investigated several approaches to developing knowledge acquisition tools to enable end users to extend a knowledge base, including analysis of Interdependency Models, scripts to plan acquisition dialogue, exploiting problem solving methods and other background knowledge, and creating English-based structured editors [Blythe et al., 2001; Kim and Gil, 2000; Gil and Tallis, 1998; Swartout and Gil 1995]. EXPECT helps users enter knowledge at the lower levels of an RHKB, and has been shown to be quite effective in several user evaluations with subjects not familiar with programming and formal languages. TRELLIS acquires more informal knowledge and is aimed to support human decision making.

The key innovative ideas behind our approach are:
- **Supporting users to create knowledge fragments from the original sources as well as from other fragments.** The key is to capture how a developer progressively generates new knowledge that results in added value to the original raw information sources. Our goal is to support users to highlight key salient information from large reports and documents, to add new knowledge fragments based on their analysis and integration of existing information, and to finally create semi-formal fragments.
- **Capturing and exploiting semantic interrelationships among information items.** TRELLIS will 1) facilitate semantic markup of relationships between different pieces of knowledge, 2) exploit semantic markups in given problem solving contexts, and 3) suggest additional relationships based on those already specified by the user. Users will be encouraged and rewarded to add valuable annotations over raw information sources, since the more annotations they add the more help the system can provide in their work. When the user chooses to do little or no annotation, the system will

4

provide weaker support (based on default heuristics and strategies) and will still help the user as much as possible.

2 **Extensible semantic markup of information items and their relationships.** Users will be able to draw from a core semantic markup language that will contain a basic domain-independent vocabulary to formulate annotations. They will also be able to extend this core language with additional terminology useful in their particular domain. Using this language, users will be able to annotate not only the information items themselves, but they will also be able to annotate the relationships among them, which will enable them to qualify and describe interdependencies between different information sources and how they relate to a new conclusion or assessment added by the developer. In essence, links between the information items will be first class citizens in the knowledge base.

Figure 1 shows an overview of the architecture of TRELLIS. A User typically starts searching the Web for a certain document, or indicating a pointer to a specific Web resource that contains useful information. Each is considered an information item. Information items may include raw information sources (an image, a text document, a video, etc.) as well as products of previous analysis (by the user or by other users.) All the information items are in some sense the knowledge base that TRELLIS operates on, and we refer to it as the Semantically Marked Information Base, or SMIB. We refer to an information item as an EI2 (Extended Information Items).

Users extend the SMIB using two tools: the Annotation Tool and the Creation Tool. They can use the EI2 Annotation Tool to add semantic annotations to an EI2 to describe its contents and to relate them to other EI2. For example, an EI2 may be annotated as containing a map, or an interesting event. The Annotation tool can also be used to relate EI2. The tool will provide an editor with a set of connectors. An example is a connector to denote that two EI2s are contradictory. This way, the user may link an EI2 that contains a description of a product as having a tag price of $20 to another EI2 that has the same product with a price of $25.

The Annotation tool draws on a library of semantic annotations and connectors that will be based on a core domain-independent language defined by the Semantic Annotation Vocabulary. An Interdependency Schema defines a vocabulary for connectors based on a variety of dimensions: pertinence, reliability, credibility, structure (x is example of y; x is part of y; x describes y, etc.), causality (x1 x2...xn contribute to y; x1 x2...xn indicate y; etc.), temporal ordering (x before y; x after y; x simultaneous with y; etc.), argumentation (x may be reason for y; x supports y; etc.). The Domain Schema contains a core vocabulary to annotate the content of documents that extends the Interdependency Schema with domain terms. Our plan is that TRELLIS will provide a core vocabulary, and users will be able to extend it with additional terms.
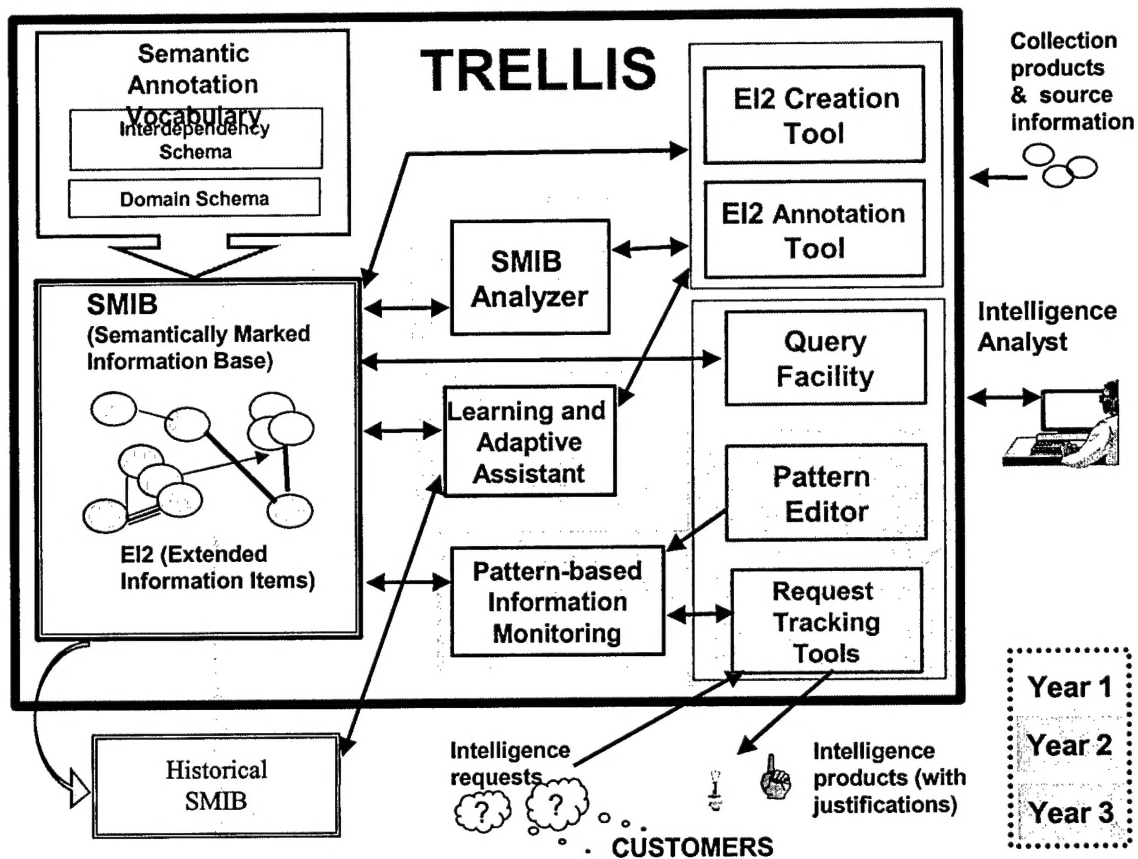
Figure 1 Overview of the TRELLIS Architecture

The Creation Tool enables users to create new EI2. For example, a user may create an EI2 as an assessment that he or she formulates based on existing EI2. If a combination of some subparts of EI2 lets a user conclude or decide on some definition, then the subparts can be captured into a new Information Item, that drops all other irrelevant parts of the original EI2. A new EI2 can be added by extracting or summarizing some of the previous results.

Figure 8 shows a snapshot of the current user interface of TRELLIS. In this case, a user is using TRELLIS to decide whether a mission to take Navy SEALs to Athens is feasible. Given the Web sources consulted and the indicated capabilities of the SEAL team (shown on the left), the user has entered the rationale for deciding that the operation is not feasible. The interface is described in more detail in a later section.

In summary, our goal is to develop tools that enable users to specify information in increasingly more structured form, and to specify semantic annotations that can be exploited for processing and integration of separate information items.

6

Figure 2 A Snapshot of the Current TRELLIS Interface

We released our first version of TRELLIS in August 2001. It includes a vocabulary for semantic annotations of decisions and tradeoffs. The initial version of this vocabulary is now available as a schema/ontology in XMLS, RDFS, and DAML+OIL. TRELLIS allows users to extend this vocabulary and its corresponding schemas. For each analysis performed by a user with TRELLIS, the system generates annotations in XML, RDF, and DAML+OIL according to the schemas and ontologies for the TRELLIS annotation vocabulary. Users can extend the vocabulary using the TRELLIS interface, adding perhaps domain specific terms or constructs that are useful to their particular task.

We exercised TRELLIS with a variety of scenarios and users to annotate tradeoffs and decisions (e.g., travel), organizing materials (e.g., search results), analyzing disagreements and controversies on a topic (e.g., political debates), and handling incomplete information (e.g., genealogy research).

The TRELLIS software is freely available under an open software license at http://www.isi.edu/licensed-sw/trellis.

Accomplishments

Our work concentrated on four major areas: 1) designing a vocabulary to annotate information analysis, 2) creating semantic markup representations for this vocabulary and annotating analysis products, 3) deriving trust ratings for sources used in collections of analyses, and 4) developing the TRELLIS interface and tools.

Each of these topics is elaborated in the sections that follow and corresponding appendices. We finalize with an overview of a future vision for this work.

2.1 An Initial Vocabulary to Annotate Information Analysis

Our initial focus application has been intelligence analysis and annotating decisions made in the presence of incomplete or inconsistent information. The main constructs that the language needs to support are to capture the following kinds of situations and for the following kinds of rationales:

- Decision making in light of inconsistent and incomplete information. This is the norm in intelligence analysis tasks.
- Information pedigree and lineage: Each piece of information comes from certain sources and may be concluded or derived from other pieces of information. Capturing this information has many implications for explanation, since we can justify the sources for each decision and users will adopt a piece of information accordingly.
- Decision making within bound resources, such as time or information availability
- Decision making over time. An assessment or decision can change in light of new information that may become available at a later time.
- Support multi-user decision making. Many users may contribute different aspects of the analysis, so it is important that the results of each user's analysis are shared by others.

Given these requirements, the desiderata for our language to annotate analysis products includes:

1) **Indicating inconsistent and complementary statements**. Assessing the truth of a statement in light of contradictory information may not be possible, or perhaps just not within the capabilities or scope of the analyst. However, it is very valuable for the analyst to annotate inconsistent statements as well as any complementary aspects of individual statements regarding a specific topic.
2) **Indicating abandoned and unexplored lines of reasoning**. The analysis process takes place within bounded resources, including the analyst's time to consider each possible alternative hypothesis or conclusion. The analyst may concentrate on aspects of the problem that seem more central and not explore other aspects that may seem secondary. Our language should help the analyst annotate options that were abandoned or unexplored for lack of time, resources, evidence, etc. For someone trying to understand the context and value of the resulting analysis it may make a big difference whether the analyst did not consider a possibility for lack of time or simply because he or she was unaware of it.

8

3) **Indicating selected statements**. The analyst may decide among alternative statements which should be dismissed and which should be pursued. Indicating which alternatives seem more promising, believable, or otherwise salient is the main added value resulting from the analysis.
4) **Indicating sources and weight of statements.** Analysts do not just consider statements at face value, but instead place enormous consideration on the sources for the statements. Based on that, they will assign believability and credibility to the source and the statements, and assess the weight that a certain piece of evidence should carry within the overall analysis.
5) **Customizable vocabulary.** Analysts in different areas may prefer certain terms to qualify statements, hypotheses, and sources. We wanted to provide an initial core vocabulary of generally useful terms that users could extend with additional terms considered appropriate for the problem at hand.
6) **Incremental refinement of qualifications**. Given the information analyzed at a given point in time, the analyst may be able to produce only a high level and possibly vague assessment that can be refined later on when more information becomes available.

We use the Dublin Core vocabulary to describe sources. The Dublin Core metadata initiative is an international effort to develop standards to describe published resources in physical or electronic form, including web pages.

We developed an initial version of the language. The constructs proposed are included in Appendix I.

2.2 Choosing a Markup Language

In order to understand the tradeoffs and differences among markup languages, we created a detailed comparison table that contrasts the different features that common languages provide. This table compares XML (eXtensible Markup Language), RDF (Resource Description Framework), and DAML (DARPA Agent Markup Language). The table is included below, and is available on-line at http://www.isi.edu/expect/projects/trellis. The on-line version of the table points to examples that illustrate the different features.

A new semantic markup language is OWL (Web Ontology Language), being developed by the W3C consortium, but the design of this language was not completed at the time that this work was performed.

We also generated and presented tutorial slide presentations on semantic markup languages, including XML, RDF, and DAML. These tutorial slides are available and can be downloaded from the project web site, http://www.isi.edu/expect/projects/trellis.

After this analysis, we decided to provide annotations for TRELLIS in all three languages. In the future, we may restrict ourselves to languages that are commonly adopted in the semantic web community.

	XML (Schema)	RDF (Schema)	DAML
Contexts	Default Namespace : xmlns Declaring others : xmlns:<label> *XMLSchema Namespace (typically labelled xsd)* xmlns:xsd="www.w3.org/2001/XMLSchema" [Final W3C Recommendation] *targetNamespace* refers to the namespace defined by the current file. <u>Example Namespace Declarations</u> **Note:** Namespaces need not point to anything in the XML Namespaces specification. <xsi:schemaLocation..> provides the location of the schema.	RDF uses XML Namespaces. *RDF Syntax Namespace* xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" *RDF Schema Namespace* xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#" Example <u>Namespace Declarations</u> in RDF **Note:** In RDF, the namespace URI reference also identifies the location of the RDF schema	DAML also uses XML Namespaces. It uses RDF & RDF Schema elements by referring to their respective Namespaces. The latest *DAML Namespace* xmlns:daml="http://www.daml.org/2001/03/daml+oil#" Example <u>Namespace Declarations</u> in DAML In DAML, we have to <u>Import</u> ontologies to be able to use the classes defined in the ontology.
Object Classes & Properties	No concept of classes and properties, only Elements of certain Types. The Type can be *simpleType* or a *complexType*.	Resource is the top level class. (http://www.w3.org/2001/01/rdf-schema#Resource) Example of <u>Classes</u>, & <u>Properties</u>	DAML also has <u>Classes</u> & <u>Properties</u>. Classes can also be a subClassOf an anonymous class created due to a '*Restriction*' on the set of all 'Things'. Two kinds of Properties are defined: *ObjectProperties* (Relate an object to another object - the value of the property is also an object) & *DatatypeProperties* (Relate an object to a primitive data type - the value of the property is a primitive data type)
Inheritance	Since there aren't any Classes, there is no concept of inheritance.	A class can be a *subClassOf* other classes (multiple inheritance is allowed)	Same as in RDF.

	However, Types can be *extended* or *restricted* , thus defining subTypes.	Properties can also be *subPropertyOf* other properties.	
Property Range	Can be specified globally. For a locally specificied (element specific) range, the element has to be declared locally. Example of a global element. The global element can be referred to like this. Example of local element	Can only be specified globally <rdfs:range....> Multiple range statements are not allowed. However the same effect can be had by having a class as the range of a property, then all it's sub-classes will be allowed as values of the Property. Example of a range specification	Can be specified globally (<rdfs:range...>) as well as locally <daml:Restriction> <daml:onProperty...><daml:toClass....> </daml:Restriction> Multiple range statements imply conjunction (i.e. all of them should be satisfied)
Property Domain	No explicit declaration of the domain of the element. The domain is implicitly the element in which the definition appears.	Can only be specified globally <rdfs:domain....> Multiple domain statements imply disjunction (i.e. any of them could be satisfied) Example of a domain spec in RDF here.	Can be specified globally (<rdfs:domain...>). Multiple domain statements imply conjuction (i.e. all of them should be satisfied) Example of a domain in DAML
Property Cardinality [1 to many relations]	Can be specified using *minOccurs, maxOccurs* Example	Not defined in the core RDF Schema. By default there are no cardinality restrictions on properties. However, new constraints like these can be specified by making them a *subClassOf* the '*ConstraintProperty*' Class. Look at how cardinality is defined in OIL-Ontology	Can be specified locally *minCardinality, maxCardinality, cardinality* Can also be specified globally although only as a UniqueProperty (single valued i.e. having cardinality of 1).
Basic Datatypes	Datatypes supported by XMLSchema are mainly variations of numerical, temporal and string datatypes. For a hierarchy of all built-in datatypes visit this link.	The core RDF Schema only includes 'Literals' which is the set of all strings. Example of the use of Literals.	Allows the use of XMLSchema Datatypes by just referring to the XMLSchema URI. Example More information can be found here.
Enumeration	Possible with the	Not possible.	Enumeration of property types is

of Property Values	*<enumeration>* tag. See an example here.		possible with the *<oneOf rdf:ParseType="daml:collection"...>* tag See an example here. It is also possible to simply point to an enumerated data type declared using XMLSchema.
Ordered Data Set	Data Sets maintain order by default **Note:** Type declarations having the *<sequence>* tag, just mean that data should appear in the order specified Example	Data Set ordered with the *<rdf:Seq...>* tag Example can be found here.	Can use the *<rdf:Seq..>* tag
Transitive Properties	Not possible to specify.	Cannot be stated.	Possible with the *<daml:TransitiveProperty>* tag See an example.
Negation	Not possible.	Not present.	Possible with the *<daml:complementOf..>* tag Example
Disjunctive / Disjoint Classes	A union of possible types for an element is possible with the *<union>* tag Example	Can use a Bag to Indicate unordered collections (or unions of properties). However, we cannot have a class as a union of 2 classes.	A Class can be a union of 2 other Classes. Possible with the *<unionOf...>* tag. We can represent disjoint unions with the *<disjointUnionOf...>* tag. Example
Necessary & Sufficient conditions for Class Membership	No	No	Yes *sameClassAs, equivalent,* using boolean combinations of Class Expressions. *UnambigousProperty* specifies a property which identifies the resource (like a primary key).

2.3 Deriving Trust Ratings Based on User Analyses

In the past, sources consulted by intelligent analysts were often authoritative and often government-controlled. Today, many sources consulted by intelligence analysts are open source (i.e., not controlled by a traditional information source). As a result, the information they provide may not be correct, reliable, and very importantly may be prone to deception. We developed in TRELLIS facilities to enable users to express their trust on a source and the statements made by it, and to combine individual views into an overall assessment of each source of information.

Our work addresses a different issue regarding whether to trust the *content* of a Web resource depending on its source. It seems that people reach some times informal consensus on how and when to trust what a source says. Many qualifiers about sources seem to be common knowledge only to those familiar with the topic. Some sources are generally considered more trustworthy or reliable than others. Some sources are considered authoritative in specific topics. Some sources are preferred to others depending on the specific context of use of the information. Some sources are considered pretty accurate but it is understood they are not necessarily up to date. Finally, specific statements by traditionally authoritative sources can be proven wrong in light of other information, while the source's reputation will still hold. In this sense, there is a finer grain of detail in attributing trust to a source with respect to specific statements made by it.

The extensions of TRELLIS that we have developed allows users to annotate the source attribution for each statement used in the analysis, to describe the source, and to make qualifications about it.

For each document indexed in TRELLIS, the user can annotate meta-data regarding its attribution using the Dublin Core [8]. The Dublin Core (*dc:*) was developed as a standard to describe resources (e.g., documents). A document is described with 15 main attributes: dc:title, dc:creator, dc:subject, dc:description, dc:publisher, dc:contributor, dc:date, dc:type, dc:format, dc:identifier, dc:source, dc:language, dc:relation, dc:coverage, and dc:rights. Five of them are concerned with attribution of information. The *dc:creator* is an entity primarily responsible for making the content of the resource. The *dc:publisher* is an entity responsible for making the resource available. The *dc:contributor* is an entity responsible for making contributions to the content of the resource. The *dc:source* is a reference to a resource from which the present resource is derived. The *dc:relation* is a reference to a related resource.

In TRELLIS, each document used in an analysis is first indexed with a short statement, as a way to summarize the particular aspect of the document used in the analysis. The statement points to the document, and must become part of an analysis unit. In the unit, the user can specify a *source* for that statement. This TRELLIS source can be any of the five fields in the Dublin Core metadata that are related to attribution and that we mentioned above, but can also be any other entity that is not indicated in it. TRELLIS

14

gives the user this flexibility because the user may trust (or distrust) any of these sources enough to take some stand about the statement.

The user can also qualify the source of a statement by its *reliability* and *credibility,* which are standard in military intelligence manuals. Reliability is typically based on credentials and past performance of the source. Credibility specifies the analyst's view of probable truth of a statement, given all the other information available to the analyst. Reliability and credibility are not the same, for example a completely reliable source may provide some information that may be judged to be not very credible given other known information. Reliability is specified by a six-valued scale ranked A to F (completely reliable, usually reliable, fairly reliable, not usually reliable, unreliable, and not possible to judge). Credibility can have one of six values on a scale (confirmed by other sources, probably true, possibly true, doubtfully true, improbable, and not possible to judge). As many users create multiple analyses that refer to common sources, TRELLIS creates an overall consensus assessment about each source as we explain in this section. We have developed an algorithm to derive and update source ratings automatically as users enter different analyses that rely on those sources. The next section shows how these ratings are used and shown to the users to help them make decisions about what sources to trust.

Trellis Web

Purpose : SEAL
Enter a Query SEAL
No. of Results 10
Conclusion : CRR time

[Main] [Logout] [Fatc]

[Add Selections]

http://excalibur.isi.edu:8888/cgi-bin/trellis_web/data.pl - Microsoft Internet Explorer

Rating Details for METOC Rota : Overall Rating of (9.8/10)

Statement	Credibility	Reliability	Used for conclusion	Tainted	Not used	Total
Avg Wind speed in March around 14 Knots :	6/6	6/6	1/1	0/1	0/1	10/10
Gibraltar avg temperature in March is 60 F :	6/6	6/6	1/1	0/1	0/1	10/10
Avg. wave height in March is 4 ft :	5/6	6/6	1/1	0/1	0/1	9.3/10
Avg Dublin temperature in March is 46 F :	6/6	6/6	1/1	0/1	0/1	10/10
Avg wave height in March is 6 ft :	6/6	6/6	1/1	0/1	0/1	10/10

NOTE: Total = (2/3)*(Cred+Rel.) + 2*Used - 2*Dis. - 1*Not.

Statement	Source	Rating
Gibraltar temperature 60 F	☐ NWS Internet Weather Source	(9.3/10) Details
Gibraltar avg temperature in March is 60 F	☐ METOC Rota	(10/10) Details
Dublin temperature is 48 F	☐ NWS Internet Weather Source	(9.3/10) Details
Avg Dublin temperature in March is 46 F	☑ METOC Rota	(10/10) Details
Water Temperature is 80 F	☑ NWS Internet Weather Source	(9.3/10) Details
Temperature must be between 90 and 50 F	☐ Unofficial Special Operations Site	(8.6/10) Details

Statement	Source
temperature	

[Search] [Import]

[Delete] [Ren] [Src Desc] [Move ->] [Cache] [Toggle]

[Add Unit] [Add Sub-Unit] [Remove] [Edit] [Import] [Import Src] [Restore] [Ex]

Figure 3. TRELLIS shows users its assessment of a source based on previous analysis by other users, showing both an overall rating and the details about how the rating was derived.

As a user is considering a topic for an analysis, he or she may wonder what sources were considered by other users on topics relevant to their analysis, as well as how those sources were rated by users in light of what they were considering and in light of their expertise on the topic. TRELLIS allows users to search for sources on specific topics, see how they rank based on their overall ratings, and view the details of a source's ratings based on the individual factors considered in deriving the ratings.

Suppose an analyst is trying to choose a drop site based on wheather conditions. Suppose at some point the analyst needs to find out the average water temperature in the Dublin area. He now invokes the "source query tool" by pressing the 'Import Src' button in the bottom right frame of the main window. Figure 2 shows the user query for "temperature" and the results that are returned. TRELLIS shows the rating of all sources that are related to the topic (here "temperature"), based on how other users refer to temperature in their analyses. The analyst then selects the sources that he considers appropriate and imports them to his selection of statements and sources in the 'Statement Editor' (bottom-left frame of the main TRELLIS window).

The user can also see further details about the ratings of a source, shown at the top of the figure. This shows the detailed factors and ratings of the source for all statements that it has been used with.

In summary, our work has concentrated on extending TRELLIS with learning and proactive capabilities. TRELLIS can capture in a finer grain the sources of each statement within an analysis and derives ratings of the trust in each source as the source is used by various users in different analyses.

2.4   TRELLIS: Helping a User Annotate Information Analysis

TRELLIS is an interactive environment that will allow users to add their observations, viewpoints, and conclusions as they analyze information by making semantic annotations to documents and other on-line resources. We view this as a knowledge acquisition problem, where users are adding new knowledge to the system based on their expertise as they analyze information.

TRELLIS has been used to annotate tradeoffs and decisions (e.g., travel), organizing materials (e.g., search results), analyzing disagreements and controversies on a topic (e.g., political debates), handling incomplete information (e.g., genealogy research), etc. In this section we use an example from an analysis for feasibility of a special operations plan to describe how users can annotate decisions with TRELLIS. Appendix II shows a screen walkthrough demonstration of how to use TRELLIS illustrated with an analysis of whether Irak used biological weapons during the Gulf War.

16

In our example, the user wants to analyze the feasibility of a mission to Athens involving a SEAL team. The mission is stated as follows[1]:

> *An FID mission for the SEALS will be to instruct the Greek Naval Special*
> *Forces to land at a designated beach, by a SEAL Delivery Vehicle (SDV)*
> *and conduct surveillance on a road junction at (38 26 05N 023 38 31E).*
> *The FID has been approved.*
>
> *A US Navy submarine with an SDV and SEAL team will depart Norfolk,*
> *Virginia (36 51 25N 076 13 41W) in early March. After a week of travel,*
> *the submarine will arrive at a port in Athens, Greece (37 55 31N 023 41(8E).*
> *SEAL team will spend several days in Athens providing instruction*
> *to Greek Naval Special Forces and on the third day, the SEAL team and a*
> *Greek Naval Special Forces team unit will be transported by submarine to*
> *an off shore area (38 21 37N 023 43 31E), where the SDV will be released*
> *with the SEAL's and Greek Naval Special Forces land at a beach (38 24(9N 023 38 09E).*
>
> *The combined team will proceed to the surveillance site and conduct*
> *three days of surveillance training on the designated road junction.*
> *The combined team will then egress to the beach landing site, rendezvous*
> *with the SDV and return to the submarine.*
>
> *The submarine will return to Athens for one day to support debriefings*
> *and a review of the training and operations.*
>
> *Once instruction has been completed, the SDV along with the SEAL team*
> *will continue routine deployment operations in the Mediterranean.*
>
> *The FID will be completed at this time.*

In this mission, the SEAL team is required to use a Seal Delivery Vehicle (SDV), which is shown in Figure 3. The SDV is like a small submarine, except there is water inside and the SEAL team is wearing diving equipment. The SDV enables the divers to approach the coast closer than they could with a bigger ship or submarine.

The user is looking into whether this mission is feasible for a given platoon. The user will check the requirements for using an SDV and check whether the location where the SEAL team is to be deployed would be adequate for using an SDV. Some of the requirements include:

- Current > 2.5 kts
- Wave Height > 3 ft combined seas
- Tides Low water < 8 ft
- Tidal range > 2 ft
- Water Clarity > 10 ft visibility from surface
- Water Temperature 50° - 60°F, wet suit; < 50°F, dry suit
- Lunar Illumination: Full moon, clear sky
- Bioluminescence: any conditions that allow visible detection submerged 10 ft in ambient light

---

[1] This example was proposed by Fred Bobbitt, a retired Special Operations officer that we had access to during this work.

The user would check the water and astronomical conditions in the Athens area. In this case, the SDV can be used in principle, but it turns out that the platoon commander recommends that the platoon does not do well in water temperatures below 65°F. Therefore, the mission is unfeasible due to the low water temperature expected in the area at that time of the year and the requirements posed by the platoon commander.

A summary of the analysis is as follows:

> Mission to Athens results in unfeasible mission for SDV

    > because conditions inappropriate

        according to source Col Dyer which is

        Very Reliable

        and Grade A Credible because Col Dyer is responsible for recommending course of action

  > conditions inappropriate is elaborated in water conditions inappropriate and lunar illumination is ok and bioluminescence is ok and tides ok

       > water conditions inappropriate is elaborated in water temperature unsust… and water currrent is ok

           > water temperature unsust... is elaborated in avg march water temperature 55-60 and platoon requires min water te...

                according to source Cmdr Bobbitt which is

                Very Reliable because Cmdr Bobbitt has 15 years exp...

                and Grade A Credible because Cmdr Bobbitt has been platoon commander for 15 years

                > platoon requires min water te... conceding min water temp requd is 50-60

                  according to source Cmdr Bobbitt which is

                  Very Reliable because Cmdr Bobbitt has 15 years exp...

                  and Grade A Credible because Cmdr Bobbitt has been platoon...

      > water current ok is elaborated in required current < 2.5 kts

```
                    because average march current 2 kts

                    according to source METOC Rota which is

                    Very Reliable

                    and Grade A Credible

        > lunar illumination is ok is elaborated in avg lun ill anticipated that week
is 30%

                    because avg lun ill anticipated that week is 30%

                    according to source Cmdr Bobbitt

        > bioluminescence is ok stands though contradicted by bioluminescence is
inadequate feb-oct

        > tides ok
```

We use now this example to describe how a user can specify this analysis with TRELLIS.

Each analysis that the user performs has a *purpose* that is used to describe the issue analyzed. TRELLIS asks the user to specify a purpose, which is a short sentence that summarizes the issue or hypothesis in question.

Each piece of information or data that is used in the analysis is called a *statement*. For example, a statement is:

> Min temp required by SDV is 50-60

A statement can be linked to a Web resource, in this case it could be a manual or authoritative source about SDV requirements, in this case:

> http://www.specialoperations.com/Navy/SDV/default.html

TRELLIS allows the user to access a Web search engine and use the results of the search to create new statements.

Users can also enter statements that are not Web pages but instead have text such as an email message, or a note about a conversation, or any other text. The user can specify a URL but it is not necessary. For example, the user could add a statement such as:

> Mission to Athens

This statement would point to the text describing the mission shown above.

In addition to input information and data, the user can add statements that indicate intermediate conclusions or hypothesis. For example:

Bioluminescence ok

indicates the user's summary of the analysis regarding astronomical data.

Statements can also be used to indicate the sources of some information. For example, if a meteorological Web site such as METOC in the Rota, Spain center is used to check the water conditions, the user can describe this source in a statement and use that to annotate the analysis. Sources can be qualified according to the user's view on their reliability and credibility (as shown in Appendix I). As we mentioned earlier, we use the Dublin Core vocabulary to describe sources.

The user can define new statements or modify existing ones at any time.

An analysis is composed of *units*, which are composed in turn of sub-units. Each unit relates individual statements using a construct from the TRELLIS vocabulary. For example, the following unit captures the analysis of water temperature requirements:

> water temperature unsustainable for SDV divers
>   ***is elaborated in***
>   average march water temperature 55-60
>   **and**
>   platoon requires min water temperature of 65
>   ***according to source***
>   Cmdr Bobbitt ***which is***
>   ***completely reliable (A)***
>   ***because*** Cmdr Bobbitt has 15 years experience with JSOC
>   ***and***
>   ***probably true***
>   ***because*** Cmdr Bobbitt has been platoon commander for 3 years

All the statements are underlined, and are always linked to some Web source or user-provided text. Users can provide reasons ("because…") or not, depending on the amount of detail that they wish to capture in this part of the analysis.

Notice that only the portions of this unit that are shown in italics and bold are part of the TRELLIS language. The rest of the components of the unit (i.e., the statements) are treated as strings and TRELLIS will not process them further.

At any time, users can compose their analysis from the statements that they have selected and created. The analysis of a purpose is made of units, which may have sub-units in turn. Users can collapse or expand any portions of the analysis and manipulate units and subunits to refine their interdependencies. Users can also rearrange the units in the analysis by dragging and dropping them in the analysis window. This is useful in cases

where the analysis is done bottom-up and the users wants to relate units that were created previously separately.

Sometimes a user does not know yet how to use a statement in an analysis but would like to have that statement included in the analysis frame. Users can select statements and include them in the analysis under a 'Notes and other information' category. This facility is also useful to drag and drop statements into the analysis.

Users can also import statements and units from other users if they are relevant to their purpose. The user can do keyword search in either purpose names or statement titles entered by other users.

TRELLIS can be used offline if the user asks the system to cache Web pages before disconnecting from the network. Statements can be selected and a cached copy of their corresponding Web pages is created and used until otherwise indicated.

Users can view the results of their analysis annotated in several markup languages: XML, RDF, and DAML, corresponding to the Schemas and ontologies that define the TRELLIS vocabulary.

2.5 Future Vision: Educating Knowledge Bases

Large knowledge bases contain a wealth of information, and yet browsing through them often leaves an uneasy feeling that one has to take the developer's word for why certain things are represented in certain ways, why other things were not represented at all, and where might we find a piece of related information that we know is related under some context. Although the languages that we use are quite expressive, they *still force knowledge into a straightjacket*: whatever fits the language will be represented and anything else will be left out. Many other things are also left out, but for other reasons such as available time and resources or perhaps lack of detailed understanding of some aspects of the knowledge being specified.

Furthermore, *knowledge ends up represented piecemeal,* compartmentalized in whatever expressions the modeling language supports. Many of the connections between different pieces of knowledge are never stated, nor can they be derived by the system given what it knows. We see no value in representing redundant information or alternative ways to deduce the same facts: if the system can derive something in one way that may be more than sufficient.

Knowledge base developers may consult many sources presenting contradictory or complementary information, analyze the different implications of each alternative belief, and decide what and how to model the knowledge. In essence, developers often capture in the knowledge base only their final beliefs about some body of knowledge. *The rationale for modeling the knowledge the way it appears in the knowledge base is not*

captured declaratively. Only consistent and complete information is captured. No indication of inconsistent but possible statements is added to the knowledge base.

As Minsky argues it [Minsky, 1970]:

> There is a real conflict between the logician's goal and the educator's.
> The logician wants to minimize the variety of ideas, and does not mind
> a long thin path. The educator (rightly) wants to make the paths short
> and does not mind -- in fact, prefers -- connections to many other
> ideas.

Knowledge base developers seem to prefer the role of logicians rather than seeing themselves as educators of intelligent systems.

Figure 4 How Knowledge Bases are Built Today

Figure 5 illustrates the limited kinds of knowledge that are captured in the final knowledge base. Developers (at least non-experts) start by consulting manuals and tutorial material, asking questions, and requesting clarifications. Their main task is to analyze and different information sources, grouping information, indexing related definitions and terms, and gathering as much raw material as possible in order to understand what needs to be represented and how. Next, they organize the information in semi-formal ways by structuring it in tables, itemized lists, and detecting opposite and complementary descriptions. Finally, they build the knowledge base itself by turning the refined results of this analysis into formal expressions that fit in the particular knowledge representation language used. Whatever documentation ends up in the knowledge base

will be the only trace left of all the design and analysis process that was done to create it. None of the documentation is captured in declarative languages. The rationale of the knowledge base design is lost: the original sources, the analysis and structuring of the knowledge therein, and the tradeoffs that were considering before deciding on the final formalization. As a result:

- It is hard to extend knowledge bases. When the knowledge base needs to be extended or updated, the rationale for their design is lost and needs to be at least partially reconstructed. The knowledge sources are no longer readily available and may need to be accessed.
- It is hard to reuse knowledge contained in knowledge bases. While it is the case that entire knowledge bases can be reused and incorporated into new systems, it is harder to extract only relevant portions of them that are appropriate in the new application. Parts of the knowledge base may be too inaccurate for the new task, or may need to be modeled in a different way to take into account relevant aspects of the new application.

In summary, knowledge has a very modest degree of mobility once it is incorporated into our current systems. Some researchers are creating shared upper ontologies that can serve as a common substrate for the more detailed knowledge in specific knowledge bases, thus facilitating interoperation and reuse. Some have argued that the brittleness in our knowledge bases can be addressed by providing common-sense and other background knowledge to these systems. These approaches may be part of the solution, but it will not address some of the issues brought up here. Current intelligent systems are hard to integrate, maintain, and understand because their *knowledge bases have not been truly educated on the topics they are supposed to know about.*

Early work on knowledge-based systems already revealed that one of the important concerns for users to accept knowledge based systems is understanding the answers they provide. Explanation generation systems were developed to describe the contents of the knowledge base as well as execution traces. Requirements engineering is very useful to describe the criteria used to define the scope and competence level of the application. Other development methodologies can be used to capture a trace of the design and software development process.

In the past few years, we have developed several large knowledge bases with EXPECT in different application areas. We found that users were interested in understanding a system's answers from quite a different perspective. What they often find most important in understanding an answer is to know <u>what sources were consulted</u> in creating that reasoning, and <u>what choices were made</u> in the presence of contradictory or missing information about certain aspects of the problem domain. Consider, for example, a system to estimate the duration of carrying out specific engineering tasks, such as repairing a damaged road or leveling uneven terrain. Users wanted to know whether common manuals and/or sources of expertise were consulted, which were given more weight, whether practical experience was considered to refine theoretical estimates, and what authoritative sources were consulted to design the content of the knowledge base.

23

We noted that they would be satisfied even if we indicated that no good source was found to describe a certain estimation and so we had interpolated from other estimates.

We believe that what was missing from our knowledge bases were the fine-grained, detailed analysis, assumptions, and decisions that knowledge engineers made in designing knowledge bases. We had not recorded in enough detail what were the original sources consulted, what pieces seemed contradictory or vague, which were then dismissed, what additional hypotheses were formulated in order to complement the original sources. As knowledge engineers we had a sense for what topics or areas within the knowledge base we were more confident about, either because we had spent more resources developing them, because we had found better sources, or because as engineers we had assessed the end result as more complete and consistent. It turns out that this information is key to put the answers of a knowledge based system in context. In other words, the analysis process that knowledge engineers perform during the implementation phase is part of the rationale of a knowledge base, and needs to be captured in order to justify answers to users. There are several other potential benefits to including such rationale within a knowledge base, such as supporting its maintenance, facilitating its integration with other knowledge bases, and transferring (and translating) knowledge among heterogeneous systems.

In summary, the goal would be to *capture the results of analyzing various information sources* consulted by knowledge engineers as they design the detailed contents of a knowledge base.

In order to empower intelligent systems, we believe we need to allow them access to the roots and rationale of the knowledge they contain. Furthermore, the knowledge base should not just contain a body of formalized expressions; rather, we should extend our view of a knowledge base to include a variety of formats and representations as well as alternative renderings of the same (or related) knowledge. They should include as many connections as possible, as stated in the original sources and as they result from the analysis of the knowledge base developer. This approach would create a new generation of knowledge bases: Resilient Hyper Knowledge Bases (RHKBs), that will be more resilient to change and reuse and will be heavier in connections and hyperlinks.

Figure 4 depicts a Resilient Hyper Knowledge. Originally, the development of the knowledge base starts with documentation, examples, dialogues (perhaps with experts), detailed explanations of special cases, notes on exceptions, hints and comments on specific topics, etc. From these, the developer will extract templates, relevant dialogue segments, itemized lists and tables to organize information, etc. This should be done while always keeping a trail of connections to the original sources. The developer will also indicate some connections between different portions of the original sources. It is our experience that many of the original sources either exist or are converted into resources on the Web, and as a result the developer can exploit the hyperlinks and connections that already exist in these original sources. As the developer continues this analysis, additional sources may be sought and incorporated at the higher levels, further enriching the grounding of the final knowledge base that is being developed.

Knowledge Base

WWW

Introductory texts,
expert hints, explanations,
dialogues, comments,
examples, exceptions,...

Info. extraction templates,
dialogue segments and pegs,
filled-out forms, high-level
connections,...

Descriptions augmented
with prototypical examples &
exceptions, problem-solving
steps and substeps, ...

Alternative formalizations
(KIF, MELD, CML,...),
alternative views of the same
notion (e.g., what is a threat)

(((
))
0)))

(defconcept
bridge ()))

Richer representations
More ambiguous
More versatile

More formal
More concrete
More introspectible

Figure 5 A Resilient Hyper Knowledge Base (RHKB)

Next, the developer can identify descriptions and associate with them prototypical examples as well as exceptions, pieces of problem solving knowledge in terms of steps and substeps, tables of parameters and values, etc. Any of these new distillations will continue to be connected to any other pieces in the knowledge base that they were derived from. The developer can also mark alternative views on the same knowledge, indicate contradictory statements by different sources, and dismiss some pieces of knowledge that may not seem useful for current purposes.

Finally, a developer can turn the more structured components into formalized expressions, in one or more languages and formalisms. Contradictory statements can be formalized and connected and marked as contradictory, for someone to pick and choose as they incorporate knowledge into a reasoning engine.

During this process, the developer can annotate the reasons for making certain decisions regarding which knowledge to model and how to model it. These annotations will help further in understanding the rationale for the development of the knowledge base.

We have described here the process with four stages to show the incremental nature of this analysis, but there may be as many levels of refinement as the nature of the knowledge and the final system may require.

Notice that in the higher levels of refinement, the representations may be richer, more versatile, but at the same time more ambiguous. In some sense, plain human language (i.e., text) may be the most mobile vehicle to state knowledge. The many users of the World Wide Web use the same pages for a variety of purposes and tasks, the ultimate signature of true knowledge reuse. At the lowest levels of refinement, the representations are more formal, more concrete, and also more introspectible, lending themselves more to automated analysis and reasoning.

There are many benefits to this approach:
- **Knowledge can be extended more easily.** The formalized, final expressions may not necessarily contain every detail in every knowledge source, but if the need arises the system is better positioned to digest additional knowledge. This could be done in two ways: the developer could incorporate the additional knowledge or perhaps the system could use some automated tools to extract that knowledge itself (since it has access to the sources and how they were originally processed).
- **Knowledge can be reused and translated at any level.** Another system can be built by reusing only the higher levels of the design process and incorporating other sources to create different final formalized expressions. Other developers can tap into any intermediate results of the analysis and do not have to start from scratch. Knowledge does not have to be reused only at the lowest level as it is today.
- **Knowledge can be integrated and translated at any level to facilitate interoperability.** Translators can be built to transform and map knowledge at higher levels. The rationale and meaning of different pieces of knowledge can be available to support translation and interoperation.
- **Intelligent systems will be able to provide better explanations.** We find that many users are reluctant to accept the solutions presented by the systems and ask for explanations not of how the system derived an answer automatically but instead ask for explanations of why the system starts out with a certain fact or belief. When users are shown the reasons for certain assumptions and the fact that certain sources were consulted to make that assumption they are reassured in the competence of the system to provide those answers. Capturing this trail within the knowledge base will enable the system to generate these kinds of justifications and explanations.
- **Content providers will not need to be knowledge engineers.** Although only those trained in the art of designing, modeling, and writing formal expressions can build the more refined portions of RHKBs, anyone can contribute to the higher levels. Many people in diverse disciplines acquire the analytical skills that suffice to organize and digest knowledge sources.

Many existing tools for text extraction (e.g, to extract significant event descriptions from news articles) and discourse analysis (e.g., to segment text into meaningful portions) could be used to support these earlier steps of the analysis. Existing approaches to derive interdependencies among pieces of knowledge may be used to help users create connections among diverse pieces of knowledge. Other tools can be developed to support transformations at the lower levels (e.g., to turn tables into instances and role values). The overhead that may be incurred in creating knowledge bases using this approach is, in our view, not as significant compared to the analysis efforts that

26

developers undergo. It may even save developers time if others can look up the rationale trail instead of asking them directly detailed questions about the portion of the knowledge base they are developing.

The approach presented here has many relations to software engineering methodologies to capture the rationale for software development, and to higher-level languages and frameworks to develop knowledge-based systems. Unfortunately, these methodologies are not common practice among developers of knowledge bases for lack of adequate tools to support developers in this process. Moreover, these methodologies are aimed at software and knowledge engineers and are not very accessible to other potential knowledge base developers, such as end users and/or domain experts.

The Semantic Web will provide an ideal substrate to ground knowledge bases into their original knowledge sources, and to contain the progressively defined pieces of knowledge and the connections among them. More and more every day, knowledge originates and ends in the Web, and we find ourselves extracting knowledge from the Web, processing it inside of a knowledge base, then putting the results back on the Web. It only makes sense to integrate knowledge bases (their content and their reasoning) more closely with the Web.

3 Personnel Supported

Faculty:
- Yolanda Gil

Research scientists:
- James Blythe
- Jihie Kim

Postdoctoral Researchers:
- Tim Chklovski

Graduate students:
- Larry Kite
- Varun Ratnakar
- Dan Wu
- Ronggang Yu
- John Lee

4 Publications

"TRELLIS: Supporting Decision Making via Argumentation in the Semantic Web", Timothy Chklovski, Yolanda Gil, Varun Ratnakar and John Lee. To appear in proceedings of the Second International Conference on Knowledge Capture (K-CAP), October 2003.

"Knowledge Mobility: Semantics for the Web as a White Knight for Knowledge-Based Systems", Yolanda Gil. In "Spinning the Semantic Web", D. Fensel, J. Hendler, H. Lieberman, and W. Whalster (Eds), MIT Press, 2003.

"TRELLIS: An Interactive Tool for Capturing Information Analysis and Decision Making". Yolanda Gil and Varun Ratnakar. 13th International Conference on Knowledge Engineering and Knowledge Management. Siguenza, Spain. October 2002.

"IKRAFT: Interactive Knowledge Representation and Acquisition From Text". Yolanda Gil and Varun Ratnakar. 13th International Conference on Knowledge Engineering and Knowledge Management. Siguenza, Spain. October 2002.

"Trusting Information Sources One Citizen at a Time", Yolanda Gil and Varun Ratnakar. Proceedings of the First International Semantic Web Conference (ISWC), Sardinia, Italy, June 2002.

"A Comparison of (Semantic) Markup Languages", Varun Ratnakar and Yolanda Gil. Proceedings of the 15th International FLAIRS Conference, Special Track on Semantic Web, Pensacola, FL, May 2002.

5   Technology Transitions

The TRELLIS concept and technology was transitioned into a new project to develop analytical tools for intelligence analysts. This project, called JIST (Just-In-caSe just-in-Time Intelligence Analysis), is building on TRELLIS to develop a new approach to intelligence analysis that provides an emerging-self-organization of knowledge regarding analysis topics and areas of interest, individual expertise, and teaming assignments. Under JIST, we are extending TRELLIS with natural language processing and machine learning capabilities to discover regularities across arguments constructed by different users.

Appendix I: An Initial Vocabulary to Annotate Decisions

Discourse Relations
  A provides background for B
  A is elaborated in B

set:members
abstract:instances
whole:part
process:steps
object:attribute
graliz:spec
A is solved by B
A shows how to do B
A is motivation for B
A depends on B
A otherwise B
A causes B
A causes choice of B
A resulted in B
A results in B
Choosing A resulted in B
A happened and resulted in B
A is purpose of B
A stands though contradicted by B (= B dismissed given A)
A conceding B
A can be interpreted through B
A evaluated by B
A restates B
A summarizes B
A in contrast with B
A . B (no relation)

Logic Connectives

Not A
A and B
A or B but not both
A or B or both
A therefore B
If A therefore B then Not B therefore Not A
If A and A therefore B then B
If Not A and A or B but not both then B
If Not B and A therefore B then Not A

Temporal Relations

A is before B
A is after B
A meets B
A is met by B
A overlaps with B
A is overlapped by B
A starts B

A is started by B
A is during B
A contains B
A ends B
A is ended by B
A equals B

Qualifiers
A definitely {not} true/false because B
A is probably {not} true/false because B
A may be {not} true/false because B
A is {not} likely because B
A is {not} impossible because B
A is {not} surprising because B
A is {not} shocking because B
A is {not} reassuring because B
A is {not} believable because B
A is {not} absurd because B
A is {not} accurate because B
A is {not} dismissable because B
A is {not} salient because B

Descriptions of Sources
Title
Creator
Subject
Description
Publisher
Contributor
Date
Type
Format
Identifier
Source
Language
Relation
Coverage
Rights

Qualifiers of Sources
Credibility
confirmed by other sources
probably true
possibly true

doubtfully true
improbable
not possible to judge

Reliability

completely reliable (A)
usually reliable (B)
fairly reliable (C)
not usually reliable (D)
unreliable (E)
not possible to judge (F)

Appendix II: A Screen Walkthrough Demonstration of TRELLIS

TRELLIS can be accessed from the main project page at
http://www.trellis.semanticweb.org, or from http:www.isi.edu/expect/projects/trellis.

Start with the Login Screen:
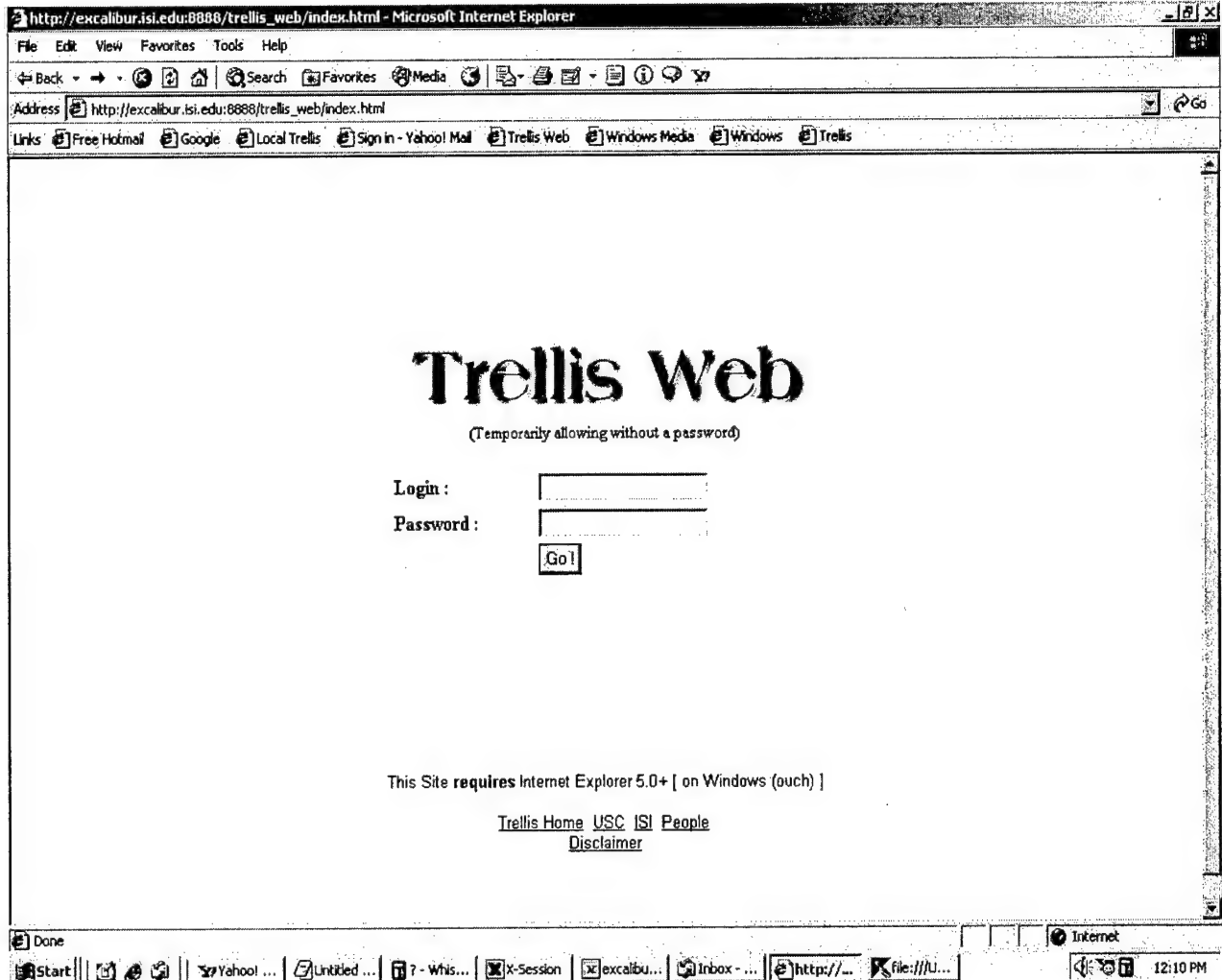
Each analysis that the user performs has a "purpose" that is used to describe the issue analyzed. After logging in, you are presented with the purposes which you have (or a person with your login ID has) made so far. By default, there are 3 purposes which are added for each new user.

Trellis Web

User: varunr

New Purpose	Save	Restore	Logout	Help	DTD	XML Schema	RDF Schema	DAML Ontology		
Should I Hire Bill Gates	Edit/View	XML	RDF	RDF Triples	DAML	Rename	Delete
Buffer Overrun Paper	Edit/View	XML	RDF	RDF Triples	DAML	Rename	Delete
Mission To Athens	Edit/View	XML	RDF	RDF Triples	DAML	Rename	Delete
Test Purpose	Edit/View	XML	RDF	RDF Triples	DAML	Rename	Delete
Which Car Should I buy	Edit/View	XML	RDF	RDF Triples	DAML	Rename	Delete
Should I Visit India	Edit/View	XML	RDF	RDF Triples	DAML	Rename	Delete
Use of biological weapons by Iraq	Edit/View	XML	RDF	RDF Triples	DAML	Rename	Delete
Test purpose	Edit/View	XML	RDF	RDF Triples	DAML	Rename	Delete
Explanation of the Bermuda Triangle	Edit/View	XML	RDF	RDF Triples	DAML	Rename	Delete

Note:
All the Markup files are kept on the server in a non web-shared directory. If the files are required to be hosted, they can be saved locally.
Any Comments or Suggestions, Please mail me

You can make a new purpose with the "New Purpose" button, which prompts for the name of the new purpose.

After creating a new purpose OR when editing an existing purpose, you end up with the main *purpose editing window* shown below. The left side of the screen is used to find, select, and edit the statements used in analyzing the purpose. The right side of the screen is used to construct and view the details of the analysis.

A *query frame* is in the top left hand side, in which you can fire off queries to a search engine (in this version of TRELLIS the search engine is Google), and get the results in the *results frame* below. There is nothing in the current selections here, since this is a new purpose. Nor is there anything in the *statements frame*, which is the frame shown in the lower left of the screen.
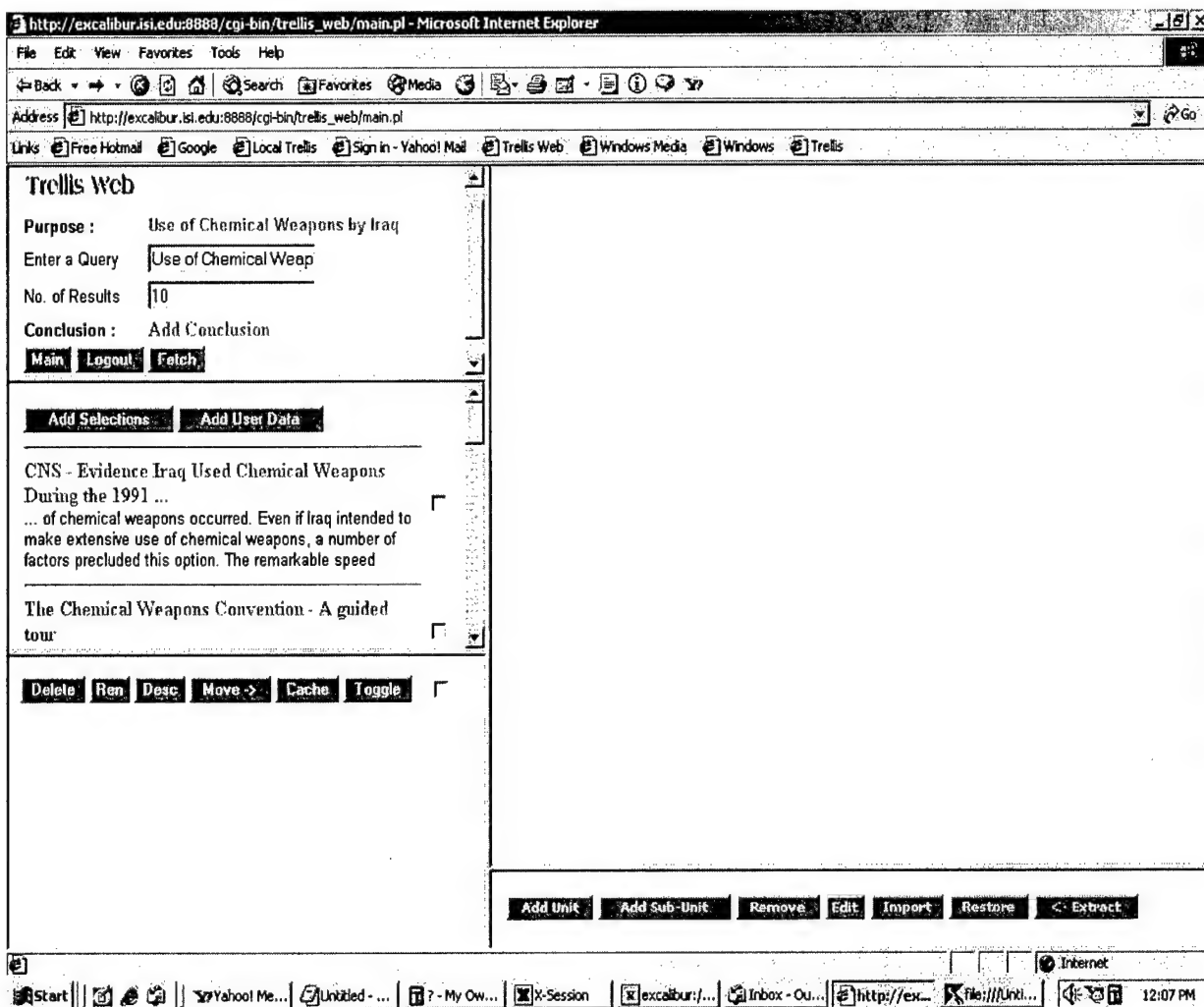
The results from the search engine that you find relevant can be selected by clicking their icon and clicking on 'Add Selections'. The Web pages selected are added to the statements frame at the bottom left.
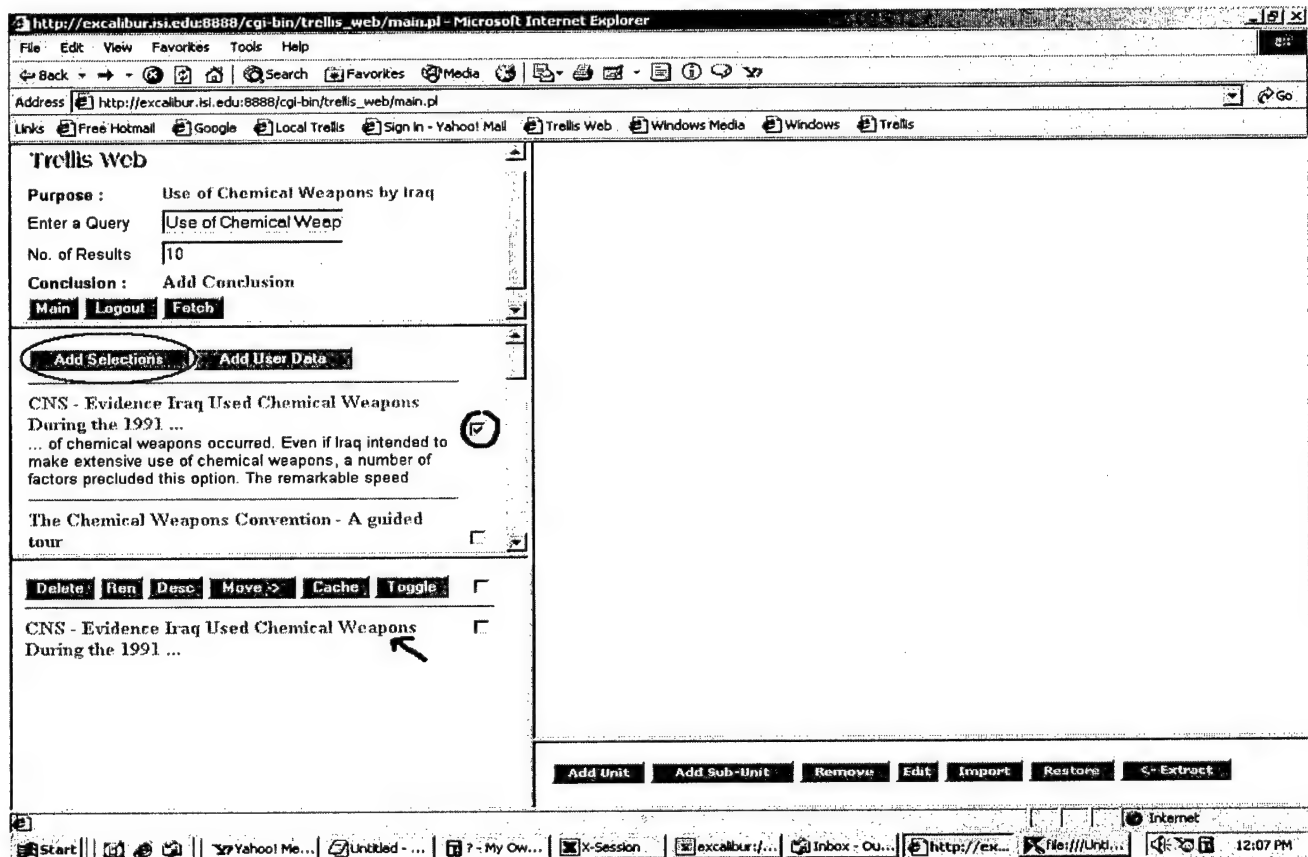
The selected pages can be renamed to reflect the salient fact that they contain that needs to be brought up in the analysis of the overall purpose. You can rename something by selecting it and pressing Ren (one of the buttons for editing of current selections). A selected page can also be deleted (selecting it and then clicking on the Delete button). Anything in the statements frame can also be moved to the analysis frame on the right hand side to start building the analysis (by selecting it and clicking the "Move ->" button).
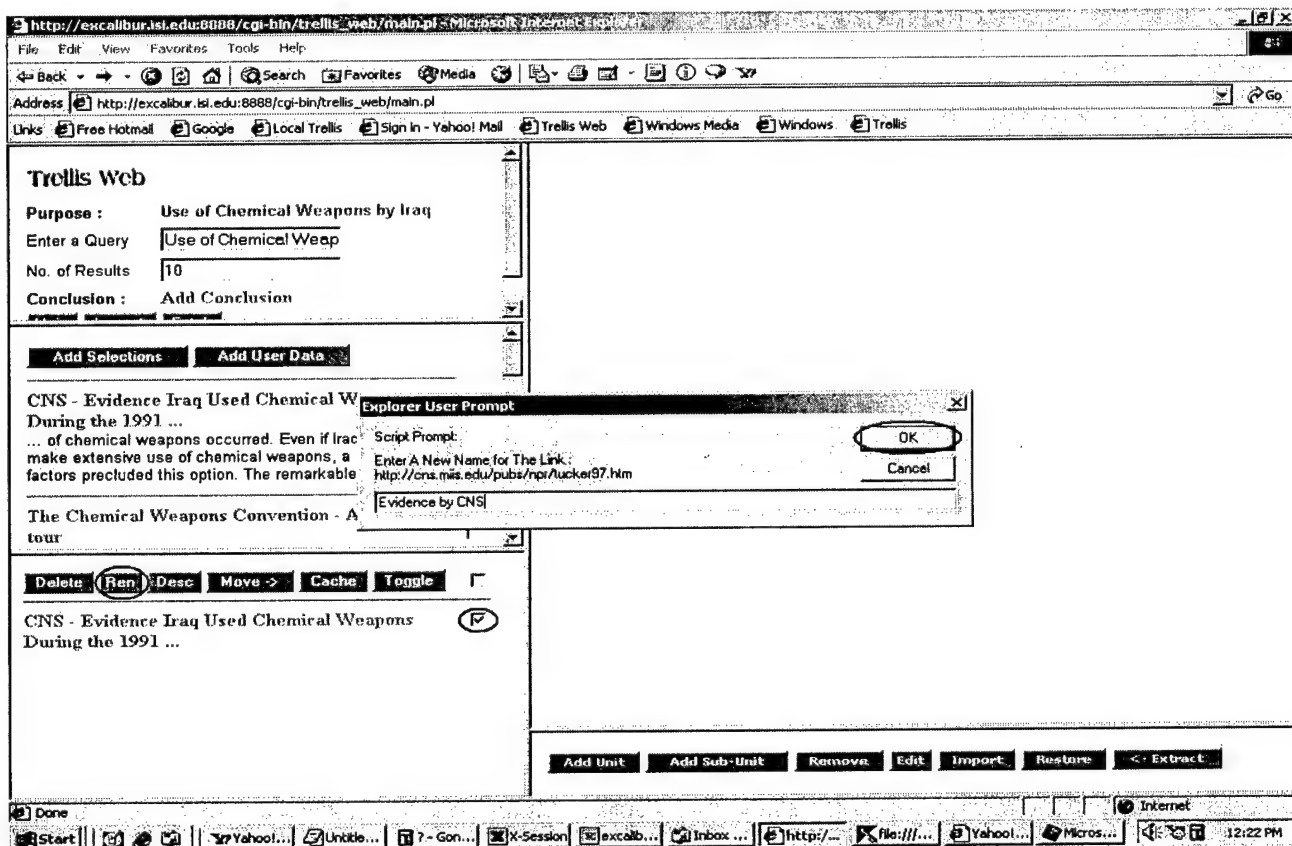
Not all statements come from Web pages. You can also add new statements manually, for example an email message, or a note about a conversation, or any other text. This is done by clicking on "Add user data". You can specify a URL if you wish, but it is not necessary.

Any new statements entered manually also have to be selected in order to be added to the statements frame and ultimately to the analysis.
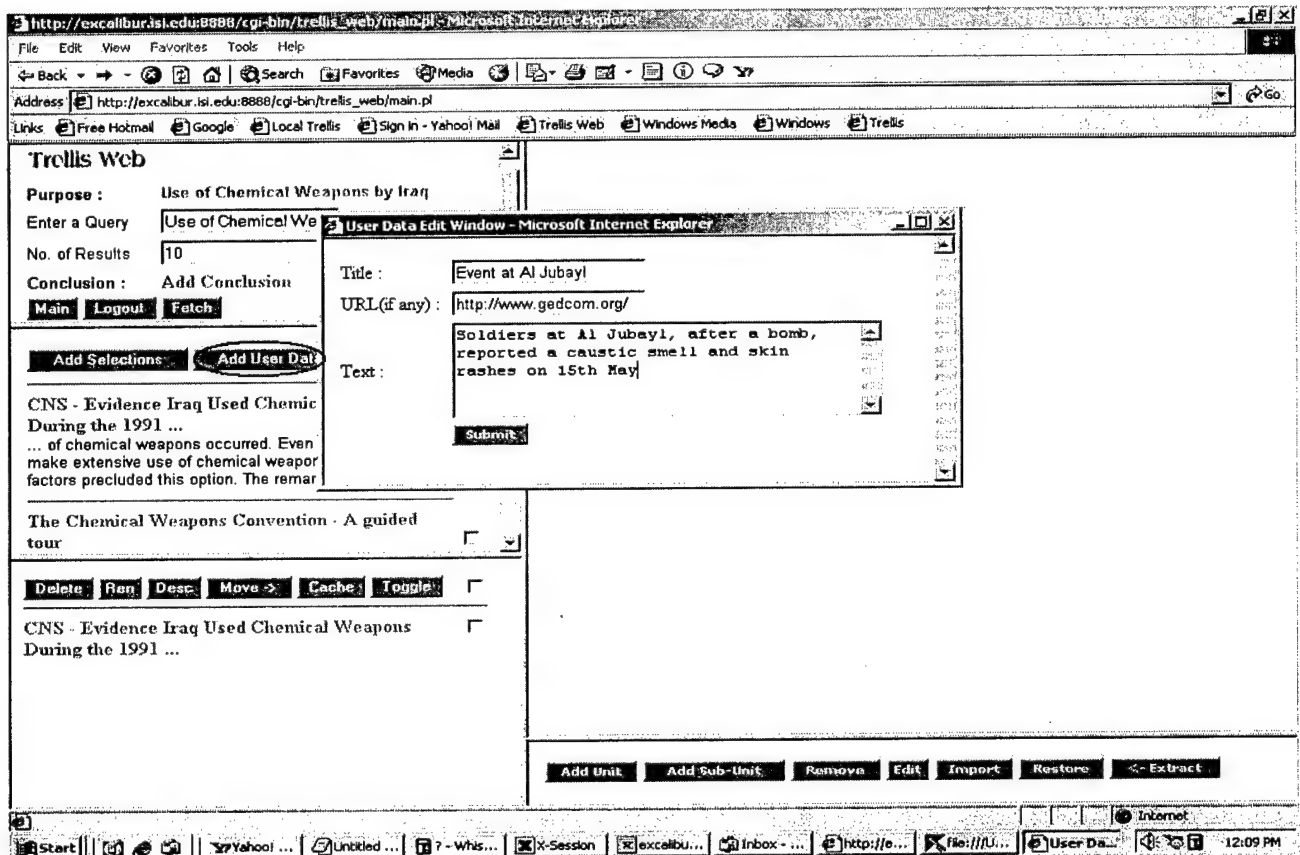
Some statements denote a source (e.g., *CNN*) that you want to cite in your analysis. You can describe a source in detail using the 'Desc' button as shown. This description will be shown whenever you click on this source statement.
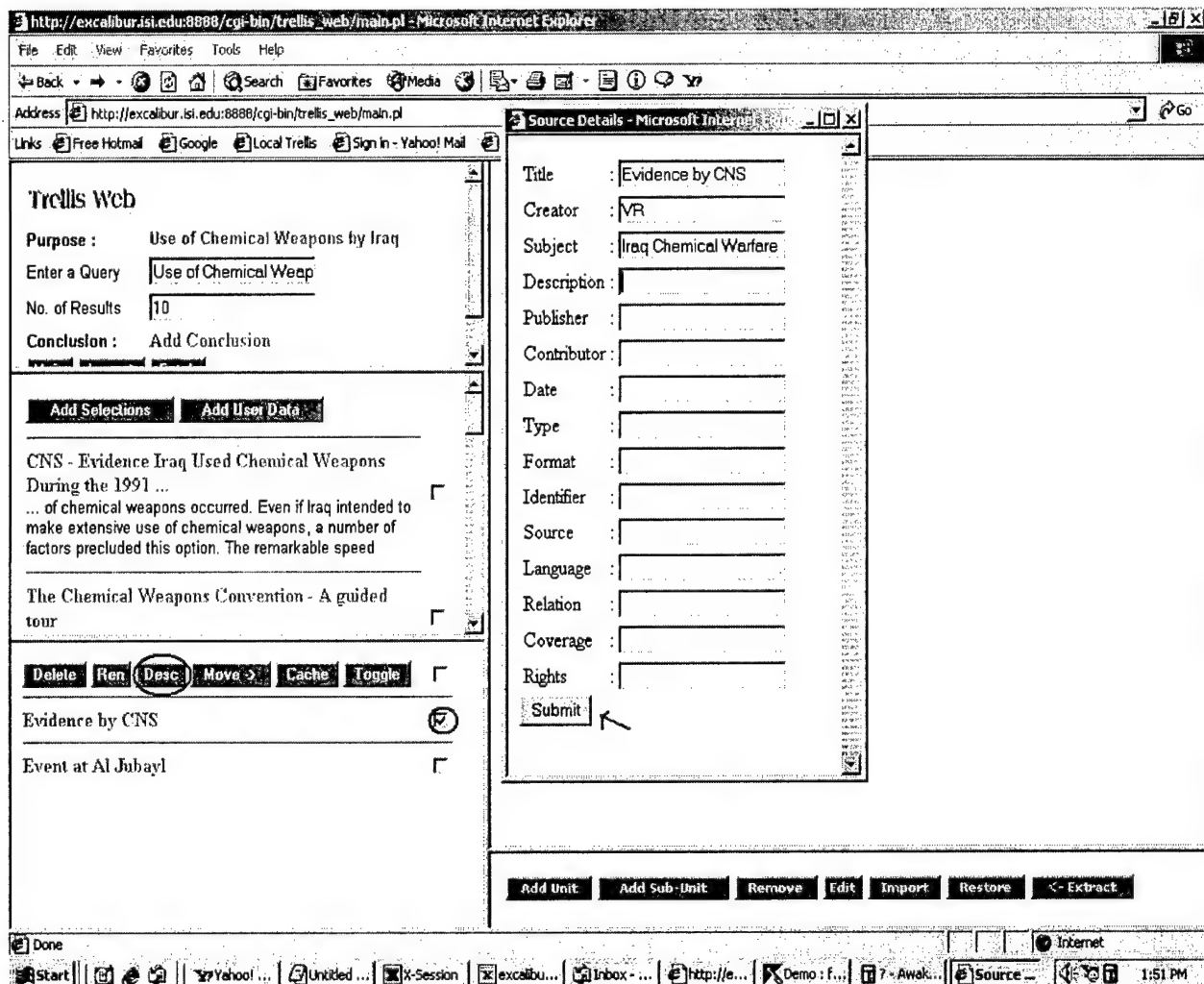
You can also view user-entered statements by clicking on them.

To modify a user-entered statement, select it and click on the Edit button, then enter the new information.
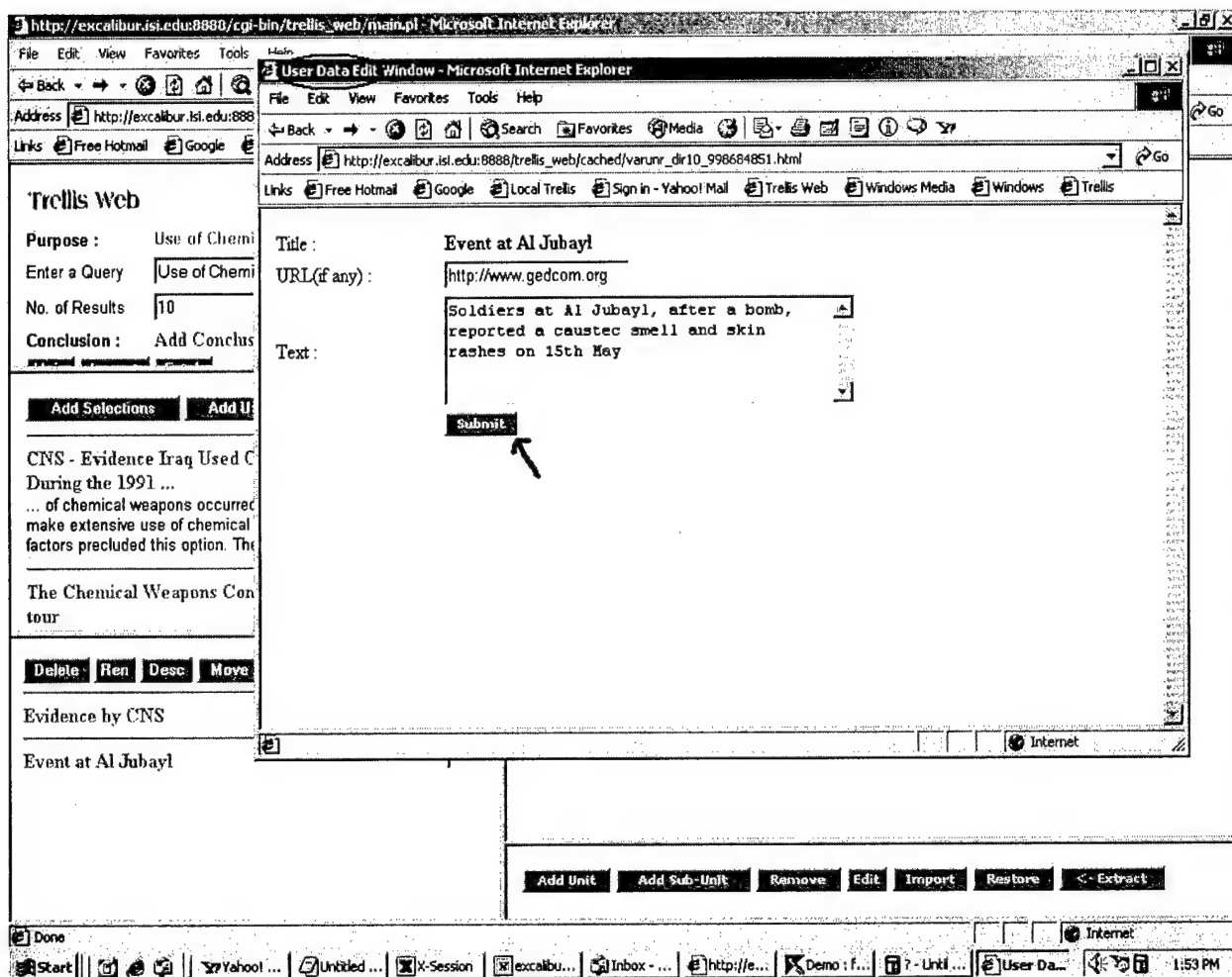
At any time, you can compose your analysis from the statements that you have selected and created. The analysis of a purpose is made of units, which may have sub-units in turn. Click the "Add Unit" button, where you can select statements and constructs from the TRELLIS language in order to create an expression. You can also annotate the source of the analysis and qualify the source for its reliability and credibility.

You can build a unit using the default TRELLIS constructs, but you can also define your own by selecting "New".

Units are normally shown as a collapsible list that the user can select and expand. If any sub-units are added after selecting a statement, they will be added under this statement in the collapsible tree.

You can also use statements and units from other users if they are relevant to your purpose. To do this, click on the "Import" button and you will be prompted for a keyword to search (here 'Iraq') in either Purpose names or Statement titles entered by other users.

From the results obtained, you can select a purpose. The system will show an uneditable version of the analysis frame. You can select any unit and import it. Notice that if you select the top unit the entire analysis is imported.

After importing, the selected units and sub-units (along with the statements used in them) as well as the sources are added to your own set.

You can also rearrange the units in the analysis by dragging and dropping them in the analysis frame. This is useful if you do your analysis bottom-up and want to relate units that you created previously separately.

To move a unit, select it first by moving the mouse to the left side of the triangle widget and clicking on it. Hold down the mouse and move it to the place where you want the unit to be. A black line shows the position where the unit would be dropped. A transparent box shows its future parent unit. (So you can either drag to, or into).

There are 2 modes of dragging : Left-click and drag drags the whole tree (as shown), and right click and drag just drags the particular unit.

After the unit has been moved:

TRELLIS can be used offline if you cache Web pages before disconnecting from the
network. Select a statement and click 'Toggle'. This link now points to a cached copy. A
link pointing to a cached copy is highlighted in a darker green color.
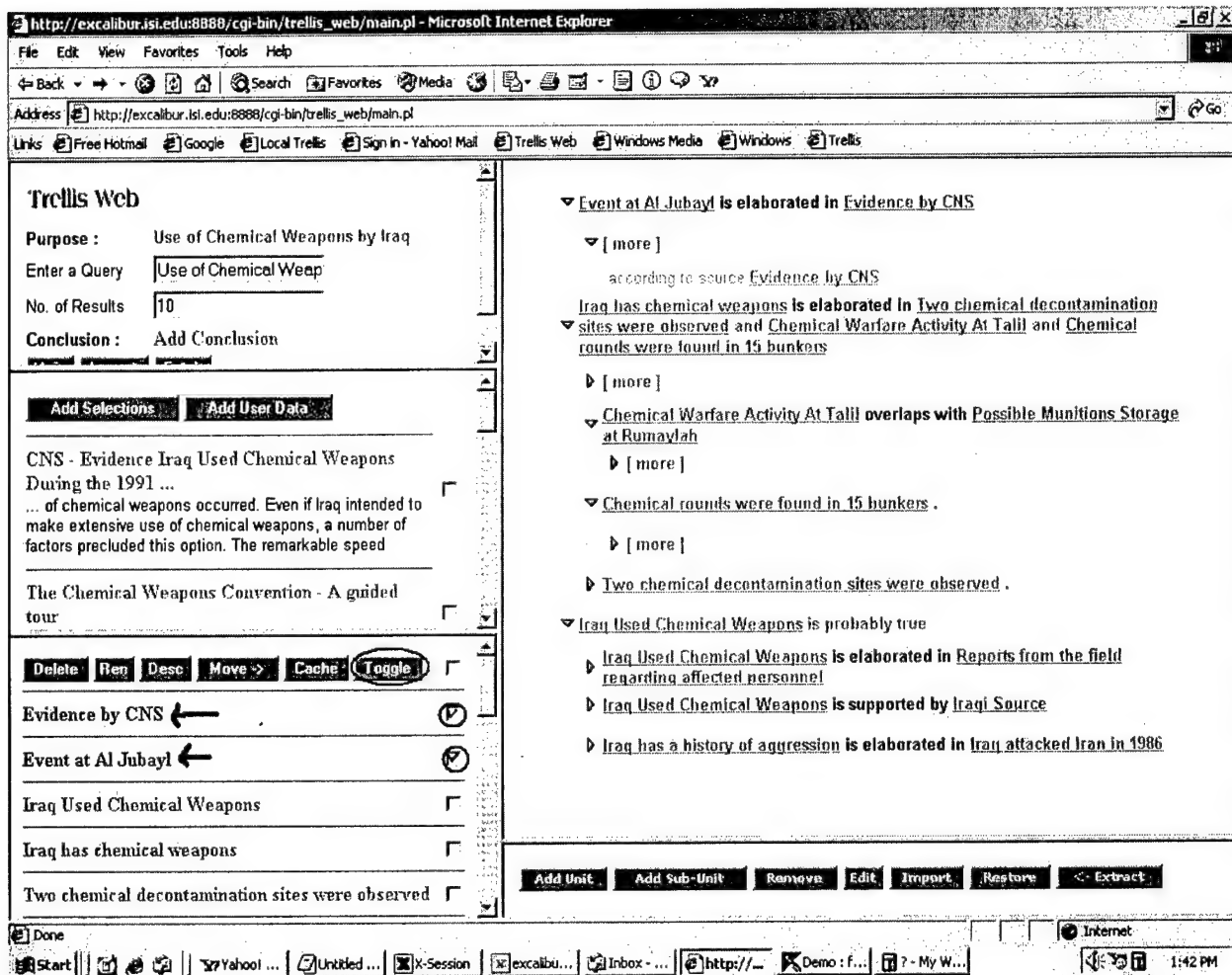
Sometimes you do not know yet how you want to use a statement in an analysis but you want to include it in the analysis frame. You can select statements and move them to the analysis frame with the 'Move->' button. These selections go under a 'Notes and other information' category.

This facility is useful to drag and drop statements into the analysis.

You can modify units in the analysis frame by clicking on the "Edit" button.

The "Remove" button deletes the selected unit and all of its sub-units.

The "Restore" button is used as a single-step undo.

The "Extract" button is used after selecting a statement. Then all the sources that it finds inside the statement are added to the current selections.

You can also view the results of your analysis annotated in several markup languages. To view this, go back to the Main Screen where you can see the marked up versions of the purposes that were just edited, by clicking on the 'XML' button for the XML markup, 'RDF' button for RDF markup and 'DAML' button for the DAML version.

In this example, the XML version of the 'Use of chemical weapons by Iraq' purpose is shown:

You can also see the Schemas (DTD, XML Schema) for the XML data, the RDF Schema for the RDF data, and the DAML ontology by clicking the respective buttons on top of the screen.

The 'Save' Button is used to take a single backup of all your purposes. If at a later time, the 'Restore' button is clicked, all the purposes from the time that 'Save' was clicked are restored.

The 'Help' Button brings up a User Guide.

The 'Logout' Button brings you back to the Login page.